THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Learning Plans by Acquiring Grounded Linguistic Meanings from Corrections

Citation for published version:
Appelgren, M & Lascarides, A 2019, Learning Plans by Acquiring Grounded Linguistic Meanings from Corrections. in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) 2019.* Montreal, Canada, pp. 1297-1305, International Conference on Autonomous Agents and Multi-Agent Systems 2019, Montreal, Quebec, Canada, 13/05/19. <http://www.ifaamas.org/Proceedings/aamas2019/forms/contents.htm#Top>

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) 2019

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

Learning Plans by Acquiring Grounded Linguistic Meanings from Corrections

Mattias Appelgren
University of Edinburgh
M.R.Appelgren@sms.ed.ac.uk

Alex Lascarides
University of Edinburgh
alex@inf.ed.ac.uk

ABSTRACT

We motivate and describe a novel task which is modelled on interactions between apprentices and expert teachers. In the task the agent must learn to build towers which are constrained by rules. Whenever the agent performs an action which violates a rule the teacher provides verbal corrective feedback (e.g. "No, put red blocks on blue blocks") and answers the learner's clarification questions. The agent must learn to build rule compliant towers from these corrections and the context in which they were given. The agent starts out unaware of the constraints as well as the domain concepts in which the constraints are expressed. Therefore an agent that takes advantage of the linguistic evidence must learn the denotations of neologisms and adapt its conceptualisation of the planning domain to incorporate those denotations. We show that an agent which does utilise linguistic evidence outperforms a strong baseline which does not.

KEYWORDS

human-robot interaction; interactive learning; knowledge representation and reasoning

ACM Reference Format:
Mattias Appelgren and Alex Lascarides. 2019. Learning Plans by Acquiring Grounded Linguistic Meanings from Corrections. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

In many commercial scenarios, workers face planning problems that consist of goal conditions that are complex and vaguely specified. For example, these problems are created by Standard Operating Procedures (SOPs)—large manuals containing rules and guidelines for workers performing complex routine tasks. In companies such as Amazon or Occado they may contain rules such as "make sure the box is properly sealed" or "never put frozen items in the same bag as meat products".

Developing a formally precise representation of such goal constraints in a planning domain, which supports inference about whether a given state complies with the goals or not, remains a major challenge. This is especially true in scenarios where the SOPs and the vast array of contingencies in which they apply are so extensive that it's untenable for a domain expert to communicate to a software developer all the ways in which the constraints manifest themselves in all possible domain states. Instead, it is more natural for domain experts to communicate their knowledge by reacting to specific situations—for example, by correcting the learner when they make mistakes. The situation where the learner made a mistake may not have come to mind to the teacher beforehand, or may result from a previous misunderstanding of the learner's capabilities.

A further challenge regarding SOPs is that they may change in unforeseen ways (this is especially true in bespoke manufacturing and in large online retail companies), making previously irrelevant domain level concepts now become relevant. For example, a company that starts to sell batteries must ensure that labels are put to the left rather than the right of the package containing them (this is a SOP in Amazon (Personal Communication)). However, the agent planning how to pack items for safe shipment may not have the domain level concept of "left" as part of its domain model (since the programmer had not initially identified these concepts as relevant).

In this paper we first present a task which is roughly analogous to, but simpler than, SOP compliant packing. In these scenarios packers must follow instructions which refer to attributes such as weight and fragility ("don't put heavy things above eggs", "protect the vase with bubble warp because it is fragile"). Our task takes place in a blocks world and we use colour as a proxy for these concepts (e.g. "put red blocks on blue blocks").

In the task we assume that agents start out ignorant of not only the goal constraints but additionally starts without a vocabulary for the terms in which the constraints are expressed (in our case, colour words), and does not have a domain model which includes these concepts either. That is, agents must learn to recognise colours from RGB values directly, not simply map words onto a symbolic language.

We present a proof of concept agent that learns to solve the task from the teacher's corrective feedback, which contains neologisms and ambiguities which the agent must resolve to decode the teacher's intended message. Working in the blocks world allows us to bypass the complex visual task of learning symbol groundings for abstract words like "heavy" and "fragile" and instead focus on how to model the *interaction* between (dynamic) symbol grounding, decoding the teacher's message, and updating goals and planning model given those messages.

We present experiments showing that a language aware agent can learn to solve the planning task in a way that outperforms a strong baseline which does not attempt to make use of the verbal content of corrections.

2 RELATED WORK

As with our task, in Interactive Task Learning (ITL), agents start out with both linguistic and operational capabilities and need to learn through interaction with a teacher. A common approach is for

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

the teacher and learner to engage in an interactive dialogue where the teacher gives instructions, definitions of words, and answers clarifying questions (e.g. [5, 17, 23, 24]).

A prime example of this is She et al [24], whose agent learns new actions from dialogue. In particular, they use a symbolic planner and goal representation to define what a specific new action should achieve. Our work extends this type of dialogue by allowing a different interaction type, namely correction, instead of just instruction and description. Another major difference is that our goal is to learn a higher level task with constraints, rather than how to perform new actions, as in She et al [24].

Other works have also tackled learning language and tasks from interaction. Wang et al. [27] learns to map language to a symbolic action language from interactions. However, the interactions assume that teacher can click through a number of possible interpretations and select the correct one. They also assume that the agent starts out with a perfect conceptualisation of the domain, while in our task, the agent starts out unaware of domain concepts that are critical to successful planning.

Knox and Stone [10] address a similar hypothesis to ours. They show with FRAMER, which is a framework for learning from "yes/no" feedback, that using human interaction to help guide an agent can make learning policies faster. This hypothesis is also shared by those attempting to learn from advice [18] which has also been shown to improve the speed of policy learning [4, 13]. However, this work mainly focuses on lower level tasks, such as motor control [10, 13], while we tackle higher level planning: finding which *sequence* of executable actions to perform.

Another method for learning from interaction is Learning from Demonstration (LfD). The majority of this work also concerns learning to perform new skills. However, a small subset of research has focused on learning plans: i.e. *when* to perform a particular action, as opposed to *how* to perform it [13, 16, 21]. A significant example is Niculescu and Matric [22] who, like us, exploit correction. The language they use is unambiguous and contains no neologisms. In contrast, in our task the language's content is hidden and must be inferred, a general and pervasive feature of natural language communication.

Reinforcement Learning (RL) [25] is another popular method for learning planning problems. The goal descriptions we are attempting to learn could be compared to learning a reward function, which is often addressed in conjunction with RL [1, 6] and approaches to learning this reward function with human interaction as evidence exist [6, 8]. However, RL is most useful for calculating expected utilites when action outcomes are stochastic. At this stage we deal with a fully deterministic domain, so we have elected not to use RL (although we may in the future).

Another significant difference from other tasks is the one mentioned earlier: that learning to ground linguistic terms involves *adapting* the domain model to include newly discovered and unforeseen concepts, rather than simply mapping terms to already known domain concepts (as in [13, 27]).

A significant part of our system is grounding language to their physical denotations. Our approach falls into a group of approaches which trains explicit classifiers for concepts [19]. Our approach shares similarity with, and was partially inspired by, the G3 framework [11, 12, 26], which builds a graphical model to represent an

Figure 1: The shades used for blocks within each colour category.

instruction. A significant difference between our work and theirs is that they concern themselves only with description, where the system seeks a correct grounding between the language and world, which we tackle correction, where a mismatch between language and world is expected, and finding what that mismatch is leads to interesting inferences.

3 THE TASK

The planning problem consists of a goal description G and a set of initial states S_0. Each $s \in S_0$ consists of 10 blocks scattered on the table. G entails that these blocks must be in a single tower, and the agent knows this. However, G also entails further constraints that the agent is ignorant of (see below). The agent experiences a state $s \in S_0$ and attempts to build a tower, receiving verbal corrections whenever it performs an action inconsistent with G, continuing in this manner until a goal state is reached. This process is repeated for each $s \in S_0$.

The constraints are rules referencing the colour of blocks. Colours can be referred to in terms of their broad colour category, such as red, green, or pink, or using the specific name of the shade, such as maroon, olive, or hot pink (Figure 1 shows the shades we use, chosen from the set of named shades in CSS3). Each rule takes one of two forms, for a pair of colours C_1 and C_2:

$$r_1^{(C_1,C_2)} = \forall x.\ C_1(x) \rightarrow \exists y.\ on(x,y) \wedge C_2(y) \quad (1)$$

$$r_2^{(C_1,C_2)} = \forall y.\ C_2(y) \rightarrow \exists x.\ on(x,y) \wedge C_1(x) \quad (2)$$

The teacher expresses both of these rules as "put C_1 blocks on C_2 blocks". Throughout the remainder of this text we will keep a running example of "put red blocks on blue blocks" which would have as potential intended message either $r_1^{(red,blue)}$ or $r_2^{(red,blue)}$ (which we abbreviate to $r_1^{(r,b)}$ and $r_2^{(r,b)}$).

The difference between the rules is which block is constrained. In $r_1^{(r,b)}$ red blocks are constrained, and need to be on blue blocks but blue blocks can have any colour placed on them. For $r_2^{(r,b)}$ the blue block is constrained but not the red. This means that for $r_1^{(r,b)}$ it is possible to build a tower with more blue blocks than red, but this is not true for $r_2^{(r,b)}$.

Focal stress in spoken language would disambiguate this intended message [15] (e.g. "put RED blocks on blue blocks corresponds to $r_1^{(r,b)}$ while "put red blocks on BLUE blocks" corresponds to $r_2^{(r,b)}$). However, in written form this is left ambiguous (and no current speech recognition system accurately estimates focal placement), so the agent must find other means to disambiguate. It should be noted that in constructing goal constraints no restrictions are made on the combination of rules allowed, so both $r_1^{(r,b)}$ and $r_2^{(r,b)}$ can be part of the goal description.

The task is implemented in a virtual environment where each scenario is defined in the Planning Domain Definition Language (PDDL) [20]. The agent can interact with the world through the action $put(x, y)$, which simply places object x on y. Further the agent only *partially* observes the current state: it can identify the blocks, their spatial relation *on* and *clear*, and it can determine the RGB-values of each block x (which we denote $F(x)$). But the agent is ignorant of how the various RGB values partition into colour concepts, as shown in Figure 1; it also starts out unaware of the available vocabulary of colour terms and must learn these from the teacher.

3.1 The Teacher's Correction Strategy

To succeed in this task our agent exploits evidence supplied by the teacher's corrective feedback. To do so the agent reasons about the teacher's dialogue strategy, which is mutually known. To simplify matters we assume that this strategy is fixed, deterministic, and correct (i.e., the teacher is sincere and competent).

The two components of the correction strategy are timing (when is a correction given?) and content (what does the teacher say?). The teacher corrects action a if a results in state s such that no sequence of actions consisting of only adding blocks to the tower would satisfy the goal G. The action a may create such a state s in two ways, as shown in Figure 2 where the rule being violated is $r_1^{(r,b)}$: either a creates a tower where the top two blocks make this rule false (state s_1 in Figure 2), or there is a block on the table which cannot be placed on the existing tower while satisfying the rule (state s_2)—in essence, there are no further blue blocks that you can add to the tower so as to put the red block on it. If we consider $r_2^{(r,b)}$ s_2 directly violates the rule while in s_1 there is no place to place the remaining blue block. This means knowing the context and the utterance is not enough to disambiguate between the intended messages.

We assume a corrective strategy where the form of the feedback discriminates whether the situation is like s_1 or like s_2 (direct violation or impossibility to place a remaining block). The verbal component of the move is the same regardless (i.e., the verbal utterance u is "No, put red blocks on blue blocks", which, as we explained before, is ambiguous between $r_1^{(r,b)}$ vs. $r_2^{(r,b)}$). When the tower directly violates the rule the teacher's multimodal move u_1 will be uttering u and pointing to the tower. If a block on the table can no longer be placed in the tower the move, u_2, is uttering u and pointing to the block (or one of the blocks) which can no longer be placed.

The agent must use the teacher's signal (u_1 or u_2) to learn to solve the planning problem. We do this by infering the teacher's

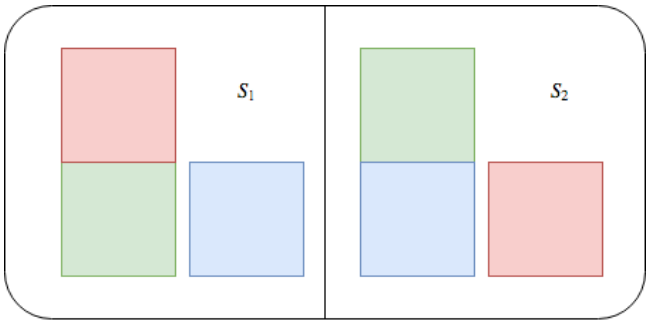

Figure 2: These two states would both be corrected if either $r_1^{(r,b)}$ or $r_2^{(r,b)}$ were in the goal.

intended message M, given u_1 (or u_2) and the knowledge that it corrects action a. If we assume the agent knows the colours of the relevant objects, it would be able to infer with certainty whether $M = r_1^{(r,b)}$ or $M = r_2^{(r,b)}$. To see this, consider s_1 in Figure 2 again and suppose the teacher says u_1: then by the semantics of correction [3]— i.e., the content of the corrective move u_1 must be inconsistent with what it corrects— given that you know the top block o_1 is red and the one beneath it o_2 is blue, then the message M that's meant by u_1 must be $r_1^{(r,b)}$ since s_1 makes $r_1^{(r,b)}$ false but satisfies $r_2^{(r,b)}$. This interaction between the context, the signal, and its meaning, given the semantics of correction, is regimented as follows:

$$Corr(a, u_1) \leftrightarrow on(o_1, o_2) \wedge (M = r_1^{(r,b)} \wedge red(o_1) \wedge \neg blue(o_2)) \vee (M = r_2^{(r,b)} \wedge \neg red(o_1) \wedge blue(o_2)) \quad (3)$$

In a similar fashion, if the speaker uses the multimodal move u_2, then the semantics of correction constrains the combination of the message M and the colours of the blocks o_1 and o_2 in the tower and the block o_3 on the table that the speaker points at:

$$Corr(a, u_2) \leftrightarrow on(o_1, o_2) \wedge (r_1 \wedge \neg red(o_1) \wedge blue(o_2) \wedge red(o_3) \vee (r_2 \wedge red(o_1) \wedge \neg blue(o_2) \wedge blue(o_3)) \quad (4)$$

Since our agent starts the learning process unable to classify the colour of any objects, it uses the above constraints (i.e., the mutually known conditions under which a corrective move is coherent) to constrain inference during learning (see Section 4.1).

4 METHOD

Our agent solves its task by jointly learning the goal constraints and learning to ground the colour terms referenced in those constraints. Figure 3 gives an overview of the agent. There are two main components: the action selection component senses the world by grounding colour terms and makes use of search and a symbolic planner to find a plan consistent with the agents current estimate of the goal description G. Correction Handling tackles learning from the teacher's corrective move by making probabilistic inferences to identify the most likely goal constraints and learn to recognise colour terms from RGB values.

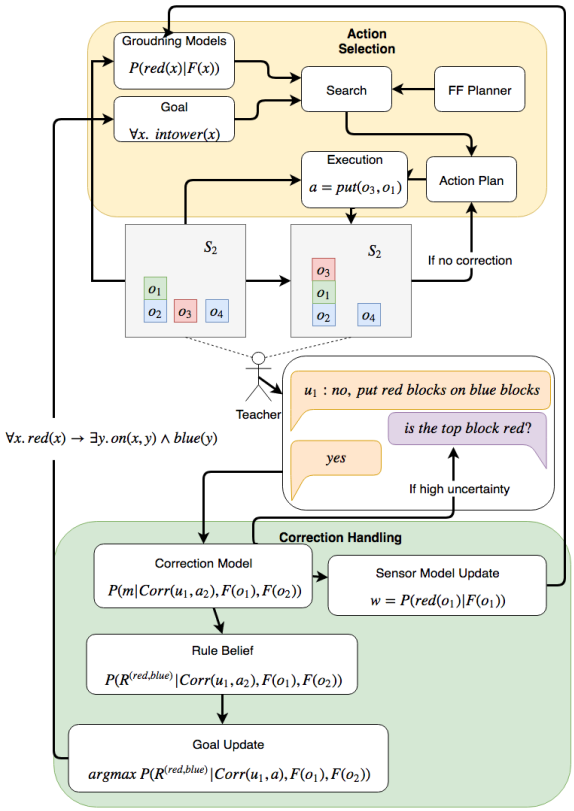

Figure 3: An overview of our Language aware agent and how its sub-systems interact.

4.1 Learning from Correction

In the event of a correction the agent generates a probabilistic model capturing the semantics of correction (Equation (3) or (4)). The variables and dependencies in the graphical model represent the relevant logical dependencies of correction semantics and therefore depend on the type of correction (i.e., is it of type u_1 or u_2?). The models, for u_1 and u_2 respectively, are shown in Figure 4.

The node $Corr(a, u_i)$ represents whether the teacher should say u_i given the colour of relevant objects and given the message. Thus

$$P(Corr(a, u_1) = True|Red(o_1), Blue(o_2), M) = 1 \quad (5)$$

if the right hand side of Equation (3) evaluates to $True$. Equivalently,

$$P(Corr(a, u_2) = True|Red(o_1), Blue(o_2), Red(o_3) \vee Blue(o_3), M) = 1 \quad (6)$$

if the right hand side of Equation (4) is True. The conditioning set are the variables that appear on the right hand side of these equations. M represents the teacher's potential intended messages: $r_1^{(r,b)}$ or $r_2^{(r,b)}$. The colour variables $Red(x)$ and $Blue(y)$ are binary variables indicating that an object is of that colour (e.g. $red(o_3)$) or not ($\neg red(o_3)$)[1]. The node $P(Red(o_3) \vee Blue(o_3))$ represents the fact that in Equation (4) o_3 is either red or blue. The remaining

[1]note: not a categorical variable saying $colour = \{red, green, blue\}$ but a binary variable indicating red vs $\neg red$ and $blue$ vs $\neg blue$ etc.

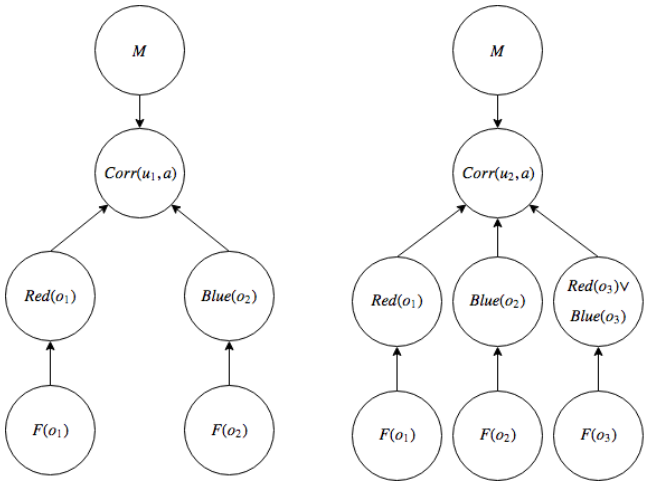

Figure 4: On the left is the correction model generated for $Correction(a_1, u_1)$: i.e. the teacher said "No, put red blocks on blue blocks" and pointed at the tower. On the right is the model for if the teacher's signal were u_2 instead: i.e. they pointed at o_3 on the table instead of the tower.

variables, $F(o_i)$, represent the visual features of the objects—i.e. the RGB values of the blocks, used since the true colour of blocks is not directly observable.

4.1.1 Correction model. The factored probability distribution the model for u_1 in Figure 4 would be:

$$P(Corr(a, u_1), Red(o_1), Blue(o_2), F(o_1), F(o_2), M) = P(Corr(a, u_1)|Red(o_1), M, Blue(o_2)) P(Red(o_1)|F(o_1))P(Blue(o_2)|F(o_2))P(M)P(F(o_1))P(F(o_2)) \quad (7)$$

$P(M)$ and $P(F(o_i))$ are set to be constant which means they will cancel out in the conditional calculations required for updating the goal (Section 4.1.3) and the grounding models (Section 4.1.5). $P(Red(x)|F(x))$ and $P(Blue(x)|F(x))$ are called the grounding models and are learned from the evidence given by the correction (Section 4.1.5).

4.1.2 Grounding. Grounding a particular colour term means accurately predicting $P(Colour(x)|F(x))$. We estimate this probability using Bayes Rule:

$$P(Colour(x)|F(x)) = \eta P(F(x)|Colour(x))P(Colour(x)) \quad (8)$$

where

$$\eta = \sum_{i \in \{0,1\}} P(F(x)|Colour(x) = i)P(Colour(x) = i) \quad (9)$$

We set the prior $P(Colour(x))$ to 0.5 since the agent has no knowledge about how likely any block is to be, for example, red or not.

We have chosen to estimate $P(F(x)|Colour(x))$ with weighted Kernel Density Estimation (KDE) (we experimented with Gaussian distributions but found them to perform poorly). Section 4.1.5 shows how this model is updated.

The factor $P(Red(o_3) \vee Blue(o_3)|F(o_3))$ is a simple extension of the grounding model, defined in terms of the estimates for $Red(x)$

and $Blue(x)$, as shown in (10):

$$P(Red(o_3) \vee Blue(o_3) = red|F(o_3)) = \frac{P(F(o_3)|Red(o_3))P(Red(o_3))}{P(F(o_3)|Red(o_3))P(Red(o_3)) + P(F(o_3)|Blue(o_3))P(Blue(o_3))} \quad (10)$$

(sharing likelihood function, $P(F(x)|Red(x))$, with $P(Red(x)|F(x))$).

4.1.3 Inferring the Goal Constraints. Given the corrections received by the teacher the agent wishes to find the most likely goal description. To begin with the agent only knows that rule constraints exist and the shape they take, further, since it does not know the colours it does not know the full space of possible rules. As the agent learns more colour terms, its hypothesis space of possible rules expands.

When the agent is corrected it uses this evidence to update the possible values of G as well as its (probabilistic) belief about which possibility is most likely. First, it extracts the possible messages M, for example, if the verbal component of the corrective move is u = "no, put red blocks on blue blocks", then the agent knows that M is $r_1^{(r,b)}$ or $r_2^{(r,b)}$. Further, whichever one the teacher did intend will certainly be part of G.

To keep track of the agent's belief about which of these rules are in the goal we introduce $R^{(r,b)}$ which represents the four possibilities of whether these rules are in the goal or not:

$$r_1^{(r,b)} \in G \wedge r_2^{(r,b)} \in G \quad (11a)$$

$$r_1^{(r,b)} \in G \wedge r_2^{(r,b)} \notin G \quad (11b)$$

$$r_2^{(r,b)} \in G \wedge r_1^{(r,b)} \notin G \quad (11c)$$

$$r_1^{(r,b)} \notin G \wedge r_2^{(r,b)} \notin G \quad (11d)$$

This variable is added when the agent encounters a correction where the potential messages have not been observed previously. That is, the first time the agent encounters u it adds this variable. The agent then keeps track of which goal constraint(s) it believes most likely through $P(R_n^{(r,b)})$ where n represents the number of corrections received with evidence relevant to $R_n^{(r,b)}$ (i.e. corrections where the possible message is $r_1^{(r,b)}$ or $r_2^{(r,b)}$).

When the teacher utters u in response to action a the observed evidence is that $Corr(a,u) = True$ and the values of $F(o_1)$, $F(o_2)$, and, optionally, $F(o_3)$ (if the teacher pointed at block o_3 on the table). For brevity we define $X = \{F(o_1), F(o_2), ...\}$ to be the observed features relevant to the correction at hand.

The agent updates its belief about $R_n^{(r,b)}$ given these observations and its previous belief $R_{n-1}^{(r,b)}$:

$$P(R_n^{(r,b)}|R_{n-1}^{(r,b)}, Corr(a,u) = True, X) \quad (12)$$

The main evidence that is relevant to this belief update is the intended message. Since this is not directly observable we marginalise over the possible messages $M = \{r_1^{(r,b)}, r_2^{(r,b)}\}$:

$$P(R_n^{(r,b)}|R_{n-1}^{(r,b)}, X, Corr(a,u)) = \quad (13a)$$

$$\sum_{m \in M} P(R_n^{(r,b)}, m|R_{n-1}^{(r,b)}, X, Corr(a,u)) = \quad (13b)$$

$$\sum_{m \in M} P(R_{n-1}^{(r,b)}|R_n^{(r,b)}, m)P(m|X, Corr(a,u)) \quad (13c)$$

To calculate $P(M = m|X, Corr(a,u))$ we use the correction model (Section 4.1.1) marginalising over the colour terms.

To calculate $P(R_n^{(r,b)}|R_{n-1}^{r,b}, M)$ we assume conditional independence between the two rules being in the goal given the message:

$$P(R_n^{(r,b)}|R_{n-1}^{(r,b)}, m) = \quad (14a)$$

$$P(r_1^{(r,b)} \in G|R_{n-1}^{(r,b)}, m)P(r_2^{(r,b)} \in G|R_{n-1}^{(r,b)}, m) \quad (14b)$$

This independence stems from the overall assumption that there is no restriction on what rules appear together. So, if $r_1^{(r,b)} \in G$ it does not change our belief about if $r_2(r,b) \in G$. The affect of m on the posterior likelihoods of the goal are encapsulated in equations (15) and (16):

$$P(r_1^{(r,b)} \in G|R_{n-1}^{(r,b)}, m \neq r_1^{(r,b)}) = P(r_1^{(r,b)} \in G|R_{n-1}^{(r,b)}) \quad (15)$$

$$P(r_1^{(r,b)} \in G|R_{n-1}^{(r,b)}, m = r_1^{(r,b)}) = 1 \quad (16)$$

To calculate $P(r_1^{(r,b)} \in G|P(R_{n-1}^{(r,b)})$ in Equation (15) we marginalise over the states of $R_{n-1}^{(r,b)}$ where $r_1^{(r,b)} \in G$, i.e., (11a) and (11b):

$$P(r_1^{(r,b)} \in G|R_{n-1}^{(r,b)}) = \sum_{s \in R_{n-1}^{(r,b)}} P(R_{n-1}^{(r,b)} = s)\delta(r_1^{(r,b)} \in s) \quad (17)$$

where $\delta(r_1^{(r,b)} \in s) = 1$ if $r_1^{(r,b)} \in s$ is true and 0 otherwise.

On expanding the hypothesis space with a new random variable (which happens at step i, say), we set the prior as in (18) and (19):

$$P(r_j^{(r,b)} \in G|R_i^{(r,b)}) = 0.01 \quad (18)$$

and

$$P(r_j^{(r,b)} \notin G|R_n^{(r,b)}) = 1 - P(r_j^{(r,b)} \in G|R_n^{(r,b)}) \quad (19)$$

One consequence of these equations is that given a correction

$$P(r_1^{(r,b)} \notin G \wedge r_2^{(r,b)} \notin G|Corr(a,u)) = 0 \quad (20)$$

which is exactly what is desired, since we know that one of the two messages must be part of the goal.

4.1.4 Updating The Goal. After a correction is given the agent updates what rule constraints are in it's goal description. Since a correction will only change the belief about rules which are potential messages of that correction this goal update can be made locally. The goal is updated using the most likely state of $R^{(r,b)}$:

$$G_n = G_{n-1} * argmax\ P(R_n^{(r,b)}) \quad (21)$$

where $G * e$ represents AGM belief revision [2]. For us, this means that when $argmax\ P(R_n^{(r,b)}) = (11b)$ then $r_1^{(r,b)}$ is added to the goal and if $r_2^{(r,b)}$ was previously part of the goal it is removed.

4.1.5 Updating the Grounding Models . To update the likelihood function $P(F(x)|Colour(x))$ there is no direct labelled data available: rather the agent must exploit predictions from the correction model to estimate how likely it is that an object is a particular colour:

$$w = P(Red(o_1) = True|Corr(a,u_1), F(o_1), F(o_2)) \quad (22)$$

we use w as a label for o_1 being red or not, thus creating a new data point $(w, F(o_1))$ for the distribution $P(F(x)|Red(x) = True)$. This probability density is estimated using a weighted KDE.

KDE is a non-parametric model which places a kernel around every known data point and calculates the probability of a point by summing over the values at that point. To take into account the weights the sum is weighted by each w and normalised by the sum of weights. For m data points $\{(w_1, x_1), ...(w_m, x_m)\}$ this becomes

$$P(F(x)|Red(x) = True) = \frac{1}{\sum_{i=1}^{m} w_i} \sum_{i=1}^{m} w_i \cdot \varphi(F(x) - F(x_i)) \quad (23)$$

Where we use a diagonal Gaussian distribution for the kernel φ.

4.2 Action Selection

To select actions we treat the task as a symbolic planning problem because the scenarios are restricted such that the agent has the requisite motor skills to perform all actions, and the outcome of an action is deterministic. We use what is learned from the steps in Section 4.1 to build the necessary symbolic description. We use the FF planner [9], a PDDL planner which requires a goal and current state description to plan.

The goal is updated as described in Section 4.1.4 and begins as

$$\forall x.\ in\text{-}tower(x) \quad (24)$$

i.e., the agent defaults to assuming no constraints on towers.

The agent can observe *on* relations and *clear* (required in the preconditions of the put action) so the only part of the state description the agent must estimate probabilistically is the colour of each block.

We predict the probability that each object is a particular colour using the Grounding Models introduced in Section 4.1.2. We define S^* to be the most likely belief state (over the colours):

$$S^* = argmax_{i \in \{0,1\}} \sum_x \sum_C P(C(x) = i|F(x)) \quad (25)$$

So if we had $P(red(o_1) = 1|F(o_1)) = 0.6$ and $P(red(o_2) = 1|F(o_2)) = 0.3$ S^* would contain o_1 as red and o_2 as not red. If we also had $P(blue(o_1) = 1|F(o_1)) = 0.7$ then S^* would contain o_1 as both red and blue, which is acceptable under our framework (this would be an incorrect inference for red and blue but it could be correct if the colours were red and maroon).

Since S^* is simply a specific belief state and since the grounding models may be incorrect it may be the case that it is impossible to build a rule compliant tower given S^* and the agent's current estimate of G. For example, if the rule $r_1^{(r,b)}$ is in the goal and there are more red blocks than blue then the planner would not be able to find a plan, since there are not enough blue blocks to place red blocks on. The agent assumes it is possible to build a tower in every scenario, so any inconsistency must be due to errors in the agents estimates.

To ensure that we do find a plan in every situation the agent performs a search over belief states, starting at S^*. The search attempts to find the most likely belief state for which there is a valid plan, maximising:

$$P(S) = \sum_C \sum_x P(C(x) = i|F(x)) \quad (26)$$

where $C(x) = 1$ if $C(x) \in S$ and 0 otherwise.

Name	Rules
One Rule	$r_1^{(red,blue)}$
Maroon	$r_1^{(red,blue)} \wedge r_1^{(green,maroon)}$
Three Red	$r_1^{(red,blue)} \wedge r_1^{(pink,red)} \wedge r_1^{(purple,red)}$
Three Rules	$r_1^{(red,blue)} \wedge r_2^{(green,yellow)} \wedge r_1^{(purple,orange)}$

Table 1: The four planning problems our agents tackled. Each problem varies in the number and identity of rules constraining the goal.

The search begins at S^* and proceeds to adjacent states by flipping the value of a prediction, for example from $Red(x) = 1$ to $Red(x) = 0$. A priority queue is kept such that the search always explores states with higher scores first.

However, to reduce the size of the search problem we use what we know about the goal constraints to only add adjacent states that move us towards states where it would be possible to find a plan. For example, if we have $r_1^{(r,b)}$ then the number of red blocks must be less or equal to the number of blue blocks. Thus, if a state has more red than blue blocks then it will be impossible to find a plan and the relative number must be changed by either increasing the number of blue or decreasing the number of red. The search method adds the highest scoring adjacent states, where one of these changes has been made. In the case where all constraints are satisfied but no plan was found the agent considers random adjacent states.

The search continues until a plan is found or a fixed number of states have been explored. In the latter case, the agent defaults to finding a random plan which builds a tower out of the remaining blocks (ensuring the agent always takes some action).

The actions in the plan are executed in sequence until either the tower is built or a correction is given by the teacher. If the teacher gives a correction the agent performs the steps in correction handling. After this the teacher resets the world state to the state which appeared before the corrected action occurred. This ensure the world is always in a state where it is possible to build a rule compliant tower without removing any blocks from the tower.

5 EXPERIMENTS

The purpose of our experiments is to test our *Language Agent* against a strong baseline which does not make use of language. We wish to show that disambiguating the message and grounding the colour terms is worth it, by showing that this agent learns faster than the baseline.

The agents must minimise regret over the training scenarios. Regret is the number of mistakes made by the agent, that is, the number of corrected actions. A perfectly executed scenario would correspond to 0 regret.

The system runs in a simulated environment and the teacher is simulated by an agent which follows the correction strategy described in Section 3.1. We run the agent through four different planning problems each consisting of 50 scenarios: i.e., 50 distinct initial sates. The planning problems vary by what rules are in the goal constraints, Table 1 details what planning problems were used. We report the regret accumulated over these scenarios.

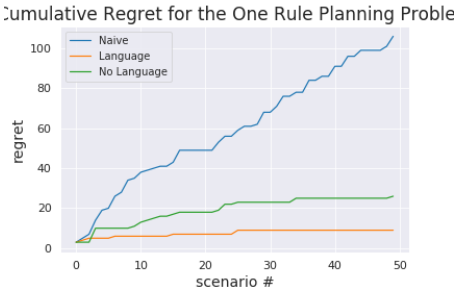

Figure 5: The regret reward for the `One Rule` planning problem.

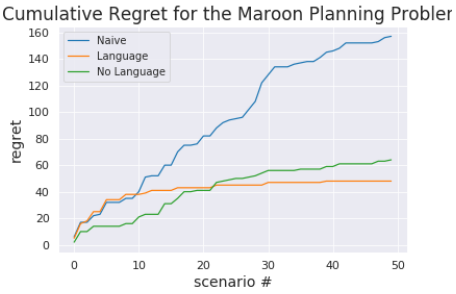

Figure 6: The cumulative regret for the `Maroon` planning problem.

Between each scenario the agent retains all knowledge it has learned so far, specifically, it retains its estimate of $P(R_n^{(C_i,C_j)})$ for all relevant C_i and C_j as well as the grounding models, $P(C_i|F(x))$.

5.1 Baselines

We compare our Language Agent to two other agents. The *Naive Agent* is an agent that does not attempt any learning between scenarios. It acts as a lowerbound on how badly an agent could perform. The Naive agent only ensures no corrected action is repeated, for example, if $put(o_1, o_2)$ was corrected the agent will not perform this action again. Otherwise the agent acts randomly. It retains no knowledge between scenarios.

The *No Language Agent*, the second baseline, attempts to learn the task without making use of the language content of the correction. It only makes use of the teacher's "no" and the pointing they do. This changes what inferences are available to the agent and therefore what it can learn from correction. Action selection stays largely the same.

Given the correction "no" + *point at tower* for action $put(o_1, o_2)$ the agent cannot infer enough to create anything resembling the rules in Equations (1) and (2) because the message doesn't convey which colour terms C_1 and C_2 are a part of the goal description (since the correction does not convey a positive example of both colours, only one). The only inference the agent can make is that blocks with similar RGB values to o_1 cannot be placed on blocks with similar RGB values to o_2:

$$\neg\exists x.y.C_{o_1}(x) \wedge C_{o_2}(y) \wedge on(x, y) \tag{27}$$

The agent adds this rule to its goal state. To identify $P(C_{o_i}(x)|F(x))$ the agent creates a grounding model with a single data point for $P(F(x)|C_{o_i} = 1)$, namely $F(o_i)$.

When the teacher points at o_3 the agent does have enough information to construct a rule along the lines of Equations (1) and (2). By pointing at o_3 the teacher is saying the block is constrained and cannot be placed any longer. This must mean it is on the left hand side of a rule and that it either must go on o_2 or o_1 most go on it. In logical form this would be:

$$r_1 = \forall x.C_{o_3}(x) \rightarrow \exists y.C_{o_2}(y) \wedge on(x, y) \tag{28}$$

$$r_2 = \forall y.C_{o_3}(y) \rightarrow \exists x.C_{o_1}(x) \wedge on(x, y) \tag{29}$$

From the available evidence the agent cannot make a decision about which rule is correct. To make an informed decision the agent will attempt to break one of the rules and observe how the teacher reacts. The agent attempts to break (28) by finding the object o_4 most dissimilar to o_3 and place it on o_2. If the teacher corrects this action then (28) is most likely the rule, otherwise it is (29).

Making use of these inferences allows the agent to build up a goal which will solve the task without disambiguating the language or performing language grounding. Between scenarios the agent retains its current estimated goal description and the distributions $P(F(x)|C_{o_i}(x))$.

5.2 Results

Figures 5–8 show the cumulative regret for each planning problem. Our Language agent outperforms the two baselines in all cases. The No Language agent performs much better than the Naive agent, showing that outperforming it is meaningful. In most problems our Language Agent has learned to act near perfectly (indicated by the the reward curve eventually going flat), while the No Language agent has only achieved this in the simplest, One Rule, planning problem. This seems to support our belief that benefiting from generalisations expressed in language outweigh the cost incurred by having to disambiguate the that language and ground the colour terms.

Further, we would expect the generalisations that language can express to be especially valuable if there is an overlap between the rules; that is, if there is a colour which is used several times in different rules, since the Language agent would be able to more quickly generalise by using the overlapping colour. We test this by comparing the Three Rules to the Three Red problem. In the Three Red problem all three rules contain "red" as one of the relevant colours. Given this, we would expect the linguistic agent to be able to more quickly learn as it does not need to learn as many colours, while the No Language agent has no way of detecting the overlap. In Figure 7, where three different rules contain the word "red", we do indeed see that there is a significant difference between the speed in which the Language agent has learned. This is especially clear when comparing to the Three Rules problem, in Figure 8. Here we see that the Language agent is having much more trouble learning the problem and is much closer to the No language agent in performance.

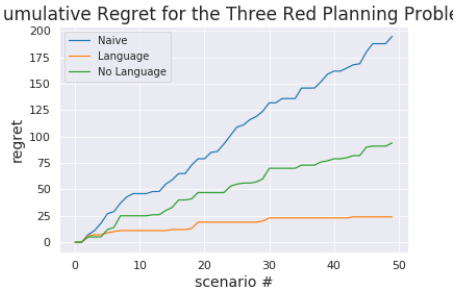

Figure 7: The cumulative regret for the `Three Red` planning problem.

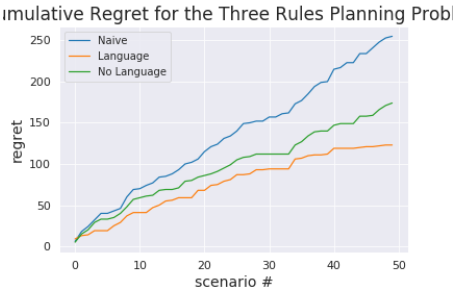

Figure 8: The cumulative regret for the `Three Rules` planning problem.

However, speedier learning is not the only benefit we get from the Language agent. The resulting domain model the agent has learned is highly interpretable. The goal description is human readable, for example, inspecting the learned goal for the One Rule problem we found that the agent has in fact learned $r_1^{(r,b)} \in G$. This is not true for the No Language agent where the goal consists of 25 different rules referencing meaningless terms such as C_{15} (which could only be made more meaningful by inspecting the data points in its grounding model). Further, since the Language agent learns colour terms these could be used for other types of communication with humans. For example, the agent could easily interpret the command "pick up a red block", given sufficient knowledge of the other words in the sentence. Therefore this would fit well with an ITL system which attempts to learn further tasks and actions.

6 CONCLUSION

In this paper we presented a novel task, where an agent learns a set of constraints from a teacher who verbally corrects the agent when it performs actions that violate the constraints. Additionally, we present an agent that explots the semantics of correction to learn the previously unknown constraints and how to ground colour terms in RGB values. Our agent consistently out-performs baseline systems which do not make use of language.

The results are encouraging, showing that using coherence relations can be a useful tool for joint task and language learning. That being said, this work represents a proof-of-concept as we make several simplifying assumptions in the domain, which makes the current approach unsuitable to be tested "in the wild".

We will discuss two main directions that must be addressed when extending to a more general setting. The world the agent inhabits and the interactions with the teacher would need to be made richer. Dealing with a richer world would require more sophisticated visual processing and reasoning about uncertain outcomes. Dealing with richer interactions requires more sophisticated ways of computing possible messages from language, a wider coverage over different types of interaction, and reasoning over ambiguity and uncertainty in the teacher's messages and strategy. To interact with real humans we would also have to be conscious of the patience of the teacher, as long wait times between interactions or the need for many repetitions would cause them to become fed up.

In the current system the main bottle-neck as far as speed of learning is learning to ground the colour terms. If we move to closer to a real world domain this is likely to be compounded as we are limited by the current state-of-the-art in visual processing, which requires models that require in the order of thousands of examples to learn. The most promising solutions to this would be to use pre-trained models and adapt them through one-shot or few-shot learning methods [7, 14], perhaps by using linguistic definitions as in [23]. We believe the method we are using to update the parameters of the grounding models is flexible enough to be integrated with a large class of visual processing systems. For this reason we are less concerned with dealing with more complex visual scenes, as we don't aim to extend the state-of-the-art in visual processing.

We are more interested in exploring richer interactions with the teacher. In our future work we plan on relaxing assumptions made on the strategy of the teacher, extend the set of constraints the agent learns, and expand the way in which the agent and teacher interact.

ACKNOWLEDGMENTS

We would like to thank EPSRC for funding Mattias Appelgren, Ram Ramamoorthy and Yordan Hristov for helpful advice and discussions, and three anonymous reviewers for very helpful feedback. All errors that remain are our own.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.
[2] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. 1985. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *J. Symb. Log.* 50 (1985), 510–530.
[3] Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation.* Cambridge University Press.
[4] Florian Benavent and Bruno Zanuttini. 2018. An Experimental Study of Advice in Sequential Decision-Making Under Uncertainty. In *AAAI*.
[5] Joyce Yue Chai. 2018. Language to Action: Towards Interactive Task Learning with Physical Agents. In *AAMAS*.
[6] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 4302–4310. http://papers.nips.cc/paper/7017-deep-reinforcement-learning-from-human-preferences
[7] Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2006), 594–611.

[8] Dylan Hadfield-Menell, Anca D. Dragan, Pieter Abbeel, and Stuart J. Russell. 2016. Cooperative Inverse Reinforcement Learning. In *NIPS*.

[9] Jörg Hoffmann and Bernhard Nebel. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. 14 (2001), 253–302.

[10] W. Bradley Knox and Peter Stone. 2009. Interactively shaping agents via human reinforcement: the TAMER framework. In *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, California, USA*. 9–16. https://doi.org/10.1145/1597735.1597738

[11] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Grounding Verbs of Motion in Natural Language Commands to Robots. In *Experimental Robotics - The 12th International Symposium on Experimental Robotics, ISER 2010, December 18-21, 2010, New Delhi and Agra, India*. 31–47. https://doi.org/10.1007/978-3-642-28572-1_3

[12] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human Robot Interaction, HRI 2010, Osaka, Japan, March 2-5, 2010*. 259–266. https://doi.org/10.1145/1734454.1734553

[13] Gregory Kuhlmann, Peter Stone, Raymond J. Mooney, and Jude W. Shavlik. 2004. Guiding a Reinforcement Learner with Natural Language Advice: Initial Results in RoboCup Soccer.

[14] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. 2011. One shot learning of simple visual concepts. In *CogSci*.

[15] Alex Lascarides and Nicholas Asher. 2009. Agreement, Disputes and Commitments in Dialogue. *J. Semantics* 26, 2 (2009), 109–158. https://doi.org/10.1093/jos/ffn013

[16] Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, and Ewan Klein. 2002. Mobile robot programming using natural language. *Robotics and Autonomous Systems* 38, 3-4 (2002), 171–181. https://doi.org/10.1016/S0921-8890(02)00166-5

[17] Peter Lindes, Aaron Mininger, James R. Kirk, and John E. Laird. 2017. Grounding Language for Interactive Task Learning. In *RoboNLP@ACL*.

[18] Richard Maclin and Jude W. Shavlik. 1996. Creating advice-taking reinforcement learners. *Machine Learning* 22 (1996), 251–281.

[19] Cynthia Matuszek. 2018. Grounded Language Learning: Where Robotics and NLP Meet. In *IJCAI*.

[20] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. PDDL-the planning domain definition language. (1998).

[21] Monica N. Nicolescu and Maja J. Mataric. 2001. Experience-based representation construction: learning from human and robot teachers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2001: Expanding the Societal Role of Robotics in the the Next Millennium, Maui, HI, USA, October 29 - November 3, 2001*. 740–745. https://doi.org/10.1109/IROS.2001.976257

[22] Monica N. Nicolescu and Maja J. Mataric. 2003. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, Proceedings*. 241–248. https://doi.org/10.1145/860575.860614

[23] Matthias Scheutz, Evan A. Krause, Bradley Oosterveld, Tyler M. Frasca, and Robert Platt. 2017. Spoken Instruction-Based One-Shot Object and Action Learning in a Cognitive Robotic Architecture. In *AAMAS*.

[24] Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Yue Chai, and Ning Xi. 2014. Back to the Blocks World: Learning New Actions through Situated Human-Robot Dialogue. In *SIGDIAL Conference*.

[25] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press. http://www.worldcat.org/oclc/37293240

[26] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth J. Teller, and Nicholas Roy. 2011. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3623

[27] Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning Language Games through Interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1224.pdf